**IT 230- Final Project Part II Submission (WPFRegisterStudent)**

**Name:** Grace Tay

**Date:** Dec. 8-11, 2016

**Class:** IT 230

**Module:** Seven

|  |  |
| --- | --- |
| **1.** | Insert a copy of your of the ZIP file of all of your Visual Studio project files here so that it can be loaded and run in another Visual Studio: |
| Insert here a copy of your \*.cs source code text you used here (copy and paste source code here, do **not** simply insert \*.cs files):  using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  using System.Windows;  using System.Windows.Controls;  using System.Windows.Data;  using System.Windows.Documents;  using System.Windows.Input;  using System.Windows.Media;  using System.Windows.Media.Imaging;  using System.Windows.Navigation;  using System.Windows.Shapes;  namespace WPFRegisterStudent  {  /// <summary>  /// Interaction logic for MainWindow.xaml  /// </summary>  public partial class MainWindow : Window  {  Course choice;  int TotalCreditHours = 0; //Used to increment how many credit hours have been registered  public MainWindow()  {  InitializeComponent();  }  private void Window\_Loaded(object sender, RoutedEventArgs e)  {  Course course1 = new Course("IT 145");  Course course2 = new Course("IT 200");  Course course3 = new Course("IT 201");  Course course4 = new Course("IT 270");  Course course5 = new Course("IT 315");  Course course6 = new Course("IT 328");  Course course7 = new Course("IT 330");  this.comboBox.Items.Add(course1);  this.comboBox.Items.Add(course2);  this.comboBox.Items.Add(course3);  this.comboBox.Items.Add(course4);  this.comboBox.Items.Add(course5);  this.comboBox.Items.Add(course6);  this.comboBox.Items.Add(course7);  this.textBox.Text = "";  }  private void button\_Click(object sender, RoutedEventArgs e)  {  choice = (Course)(this.comboBox.SelectedItem);  string courseName = choice.ToString();  switch (validateUserSelection(choice))  {  case 0://Display error confirmation: Already registered  label3.Content = "You have already registered for this course " + courseName;  break;  case 1: //Display error confirmation: Too many credit hours  label3.Content = "You cannot register for more than 9 credit hours.";  break;  case 2:  choice.SetToRegistered(); //Sets registration bool to true (See ValidateUserSelection function)  listBox.Items.Add(choice); // Display a registration confirmation message  label3.Content = "Registration confirmed for course " + courseName;  TotalCreditHours += 3; // update the total credit hours textbox if registration is confirmed for a selected course  textBox.Text = TotalCreditHours.ToString();  break;  }  }  private int validateUserSelection(Course selectedCourse) //Create code to validate user selection (the choice object)  {  if (selectedCourse.IsRegisteredAlready()) //Checks to see if course is already registered  {  return 0;  }  else if (TotalCreditHours > 8) //Checks to see if exceeded 8/ reached 9 credit hours  {  return 1;  }  return 2;  }  }  } |
| **2.** | Insert a screenshot here of the output that resulted from running your program, showing your last name as the first printed text to the screen: |
| **3.** | Explain the design of your program, the steps you took to complete it, and how you coded it:   1. I realized when I first did the program (see end of document) I went the long way around and did everything incorrectly, since there is a publically declared Boolean checking for registration. I restarted the project with this information in mind. 2. The first bit of code is a separate private int function, validateUserSelection with the choice object eventually passed in as a parameter.    1. validateUserSelection returns 0 if the course object’s public IsRegisteredAlready function returns a true Boolean.    2. Else if the TotalCreditHours exceeds 8/ reaches 9 credit hours [note, this is fixed based on prior instructor feedback] the function returns a 1.    3. If neither of these conditions are true, then the function returns a 2. 3. In the button\_Click void function, I first convert choice to a string and save it as courseName. 4. Then, the main bit of code is a switch statement with validateUserSelection(choice) as the controllingExpression.    1. If the controllingExpression returns a 0, we go to the case constantExpression 0, which tells the label3 text to update to “You have already registered for this course “ plus the course name.    2. If the constantExpression is 1, the text updates to “You cannot register for more than 9 credit hours.”    3. Lastly, if the constantExpression is 2, we call the Course object, choice, and tell it to SetToRegistered and change the isRegisteredAlready Boolean to true. I Add the choice to the listBox and update the label3 text to say “Registration confirmed for course “ plus the selected course name. Then, I increment the TotalCreditHours by 3 and update the textBox to reflect the number of credit hours. |
| **4.** | Reflect on this experience and the lessons you learned from it:  My biggest mistake was skimming the Course.cs file without realizing the registration Boolean. As you can see below, I did this program inaccurately which caused me to spend many hours on an inefficient program. I think I thought too deeply into the past assignment trying to take everything literally without trying to apply the concepts taught. Once the program was finished, it was pretty easy to see this from a whole view. I describe the functionality below. I think since I can say what the program is doing from a big picture, that is a big lesson learned.  The click function is called each time we press the button to add a course. With each click, we are going to call that validateUserSelection passing in the choice selected. I check to see if that course has been registered already, which will definitely be false the first time through. Since that is the case, we move to the next bit checking to see if the TotalCreditHours exceeds 8, also false, so the function returns a 2. That 2 is passed into the switch, which takes me to case 2. This allows the course object’s Boolean function to be called which changes the bool to true for IsRegisteredAlready. This case also adds the registration, tells you what you’ve registered for, and then increments TotalCreditHours and you can see in the textBox the number of credit hours Now, when you click again on the same course, we hit that first if statement and it returns a 0, which brings you to the switch statement and you get an already registered error.  This assignment brought in various elements from past assignments: switch, if/else, ToString(), and it was also important to keep an eye on access levels. At one point, my entire program was wrong because I initialized TotalCreditHours in the button click. If you think about what that is doing, you basically reset the TotalCreditHours to zero each time you click. Once I initialized TotalCreditHours on the class, everything worked. It is amazing how initializing or calling something in the wrong spot can cause the whole program to be out of whack.  **Original Program (Worked, but wasn’t as efficient)**  The first step was reading through the assignment guidelines, based on that I knew I eventually had to focus on the button\_Click() event handler.  After perusing the assignment, I opened up the program and ran it to check how well it was working and make sure all the visual elements are there. While it looks pretty, you cannot even register a course.  Once I got back into the code, I navigated to Solution Explorer, clicked on MainWindow.xaml.cs file and clicked on the button\_Click (), which brought me to the “To Do section”.  First, I do the simpler task of just confirming when you’ve registered four a course by updating the label3 Content and then Add choice to the listBox.    Now, I get into the actual code. So, my first assumption is that the code we learned is somehow supposed to be relevant to this assignment. For this reason, I converted each course name into a respective integer identifier (1-7). I then reused a lot of code from the console counterpart to this assignment.  I had two separate functions, an IdentifyCourse integer function uses a switch statement to take the string value of comboBox and convert it a numericalCourseIdentifier.  I have a second function called ValidateUserSelection which takes the parameters int choice, firstChoice, secondChoice, thirdChoice, totalCredit. I reuse a lot of the code, exept for the return -3. I did this wrong previously, so I changed it to totalCredit > 8 instead of = 9. Everything else checks to see if (1) the choice is out of bounds [this isn’t really possible in this scenario], (2) if this choice has already been made, (3) if you’ve exceeded the maximum credit hours, or (4) if you can register for the class.  Then, inside the button click, I allow the selection of course items and convert it to a string. I Identify the course and have it converted to an integer representative. Then, I have a switch case which takes the data from my ValidateUserSelection and brings it into a switch statement. Depending on the case, the label3 text content will reflect the selection made and allow the user to know if they have already registered or if they cannot register. If registration is confirmed, then the listBox will reflect the course added and store it inside of either firstChoice, secondChoice, or thirdChoice. Here, the TotalCredits is incremented by 3 and the textbox for total credits receives a string representing how many credits have been registered.  I am going to show the code as a screenshot here. Now that everything runs successfully, I am going back to optimize the code where possible. The final solution will be written above. |
|  |  |